INF600A: Laboratoire #4
Utilisation, dans un style **fonctionnel**, des méthodes du module `Enumerable`

Jeudi 25 octobre 2018
13h30–15h30
PK-S1565

> *Learn to use* `Enumerable`. *You will not be a rubyist until you do.*
>
> *«Ruby QuickRef», R. Davis (http://www.zenspider.com/Languages/Ruby/QuickRef.html)*

Le but de ce laboratoire est de vous familiariser avec l'utilisation des méthodes du module `Enumerable` de Ruby, donc principalement avec la partie de Ruby qui permet l'utilisation du style de **programmation fonctionnelle**.

Pour obtenir les fichiers pour ce laboratoire :

```
$ git clone http://www.labunix.uqam.ca/~tremblay_gu/git/Enumerable.git
```

Ce qui est fourni

- `array.rb` : Une **extension** de la classe `Array`, qui définit diverses méthodes pouvant être appelées sur un tableau, **méthodes que vous devez mettre en oeuvre.**
- `array_spec.rb` : Des tests unitaires, écrits avec `MiniTest`, pour chacune des méthodes que vous devez mettre en oeuvre.
- `makefile` : Un fichier qui permet d'automatiser l'exécution des tests unitaires, et ce en définissant une (1) cible par méthode — en plus d'une cible globale `tests` pour exécuter tous les tests.

  Pour tester une méthode spécifique, il suffit de changer la variable `TEST` dans le `makefile` — voir page suivante.

Plus spécifiquement, chaque méthode à mettre en oeuvre est précédée d'un commentaire qui donne une brève **description** de ce que fait la méthode. Cette description est suivie d'un **petit exemple**, simple. Pour des exemples plus détaillés et plus complets, il faut consulter les tests pour la méthode, dans le fichier `array_spec.rb`.

Exemple : la première méthode à mettre en oeuvre est la méthode `pairs`

- Fichier `array.rb`

```
# Retourne les elements pairs du tableau.
#
# Exemple:
#   [10, 23, 30].pairs == [10, 30]
#
def pairs
end
```

- Fichier `array_spec.rb`

```
describe "#pairs" do
  it "retourne [] quand []" do
    [].pairs.must_equal []
  end

  it "retourne le tableau quand singleton pair" do
    [10].pairs.must_equal [10]
  end

  it "retourne [] quand singleton impair" do
    [11].pairs.must_equal []
  end

  it "retourne le tableau quand juste des pairs" do
    a = 100.times.map { 2 * Integer(rand) }
    a.pairs.must_equal a
  end

  it "retourne [] quand juste des impairs" do
    a = 100.times.map { 2 * Integer(rand) + 1 }
    a.pairs.must_equal []
  end
end
```

- Fichier `makefile` :

```
# Il suffit de changer la variable qui suit par un autre nom de
# methode pour executer un autre test.
TEST=pairs

################################################################
....

pairs:
	ruby array_spec.rb -n /pairs/
....
```

Si vous désirez exécuter les tests pour la méthode `foo`, deux possibilités :

a. Dans le `makefile`, vous modifiez la ligne «`TEST=pairs`» par «`TEST=foo`», puis vous exécutez simplement la commande «`make`».

b. Vous lancez la commande suivante au niveau du *shell*, sans modifier le `makefile` :

```
$ make TEST=foo
```

Et si vous voulez exécuter les tests pour l'ensemble des méthodes :

```
$ make TEST=tests
```

Pour l'ordre à suivre pour la mise en œuvre des méthodes, suivez l'ordre des cibles dans le fichier `makefile`.